Searching

- **Search**: locate an item in a list of information

- **Linear search**:
  - Starting at the first element, this algorithm sequentially steps through an array examining each element until it locates the value it is searching for.
Searching

- Search function arguments:
  - list (array)
  - size (number of elements)
  - value being searched for
- What should the function return?
  - bool?
  - the found item? (maybe it’s a structure)
  - position in list?

Searching

- Search function will return the position:
  - use -1 for not found (it’s not a valid index)
  - the calling function can use the position to access the found item (in case it’s a structure)
  - the calling function can use the position for other purposes, such as deleting or moving the item in the list.
• **getBookPosition:**

// getBookPosition: accepts a list of books, the number of books // in the list, and the title of a book to find. // Searches list for occurrence of title, records its position // If it's not in the list, returns -1, otherwise position.

```cpp
int getBookPosition (string list[], int numElems, string title) {
    int position = -1;     // position of title in array, // -1 => not found yet
    for (int i=0; i<numElems; i++) {
        if (list[i]==title) {
            position = i;  // change position only when title found
        }
    }
    return position;
}
```

---

**Problems with getBookPosition**

• How many times does the loop repeat?

• What if the title is in the list more than once, which position is returned?

• How can we make the loop stop as soon as it finds the value its looking for?
From Assignment 7

• getBookPosition, revised:

```c
int getBookPosition (string list[], int numElems, string title) {
    int position = -1; // position of title in array
    bool found = false; // flag to track when title is found
    int i=0;
    while (i<numElems & & !found) {
        if (list[i]==title) {
            found = true;
            position = i; // change position only when title found
        }
        i++;
    }
    return position;
}
```

Book Inventory Example

• Goal: use an array of structures to represent a bookstore inventory

- Information about a book
  - `sku`: (stock keeping unit) unique for each book
  - `title`
  - `quantity` (number in stock)
Book Inventory Example

• In C++:

```c++
// global
struct BookEntry {
    int sku;
    string title;
    int quantity;
};

// inside main function:
const int MAX_INVENTORY = 10000;
BookEntry inventory[MAX_INVENTORY];
```

Search function for Book Inventory

• Find a book with a given sku
• Search function parameters:
  - list (the inventory)
  - numElems (number of elements)
  - sku
• Return the position of the bookEntry with the given sku, or -1 if not found.
Book Inventory Search

- **findBookWithSku:**

```c
int findBookWithSku (BookEntry list[], int numElems, int sku) {
int position = -1;    // position of bookEntry in array
bool found = false;   // flag to track when book is found

int i=0;
while (i<numElems && !found) {
    if (list[i].sku==sku) {
        found = true;
        position = i;  // change position only when title found
    }
    i++;
}
return position;
}
```

- **Using findBookWithSku**

```c
int main {
const int MAX_INVENTORY = 10000;
BookEntry inventory[MAX_INVENTORY];
int numElems = 0;

getInventory(inventory,numElems); // input inventory (file?)
int sku = getSku();               // input sku from user

int index = findBookWithSku(inventory,numElems,sku);

if (index==-1)
    cout << “No book in inventory with that sku” << endl;
else {
    cout << “sku:” << inventory[index].sku << endl;
    cout << “title:” << inventory[index].title << endl;
    cout << “quantity:” << inventory[index].quantity << endl;
}
```

Assume these functions are defined elsewhere in the program.
Search for a given title

- How would you change `findBookWithSku` so that it would find a `BookEntry` with a given title?
  - call it `findBookWithTitle`
  - what else?